<https://www.java67.com/2015/10/2-ways-to-find-duplicate-elements-in-java-array.html>

**2 Ways to find duplicate elements in an Array - Java**

Problem: You have given an array of objects, which could be an array of integers and or array of Strings or any object which implements the Comparable interface. How would you find duplicate elements from an array? Can you solve this problem in O(n) complexity? This is actually one of the frequently asked [coding problems from Java interviews](http://javarevisited.blogspot.com/2015/10/133-java-interview-questions-answers-from-last-5-years.html). There are multiple ways to solve this problem and you will learn two popular ways here, first the brute force way, which involves comparing each element with every other element and other which uses a hash table like data structure to reduce the time complexity of problem from quadratic to linear, of course by trading off some space complexity. This also shows that how by using a suitable data structure you can come up with a better algorithm to solve a problem. If you are preparing for programming job interviews, then I also suggest you take a look at [Cracking the Coding Interview](http://www.amazon.com/dp/098478280X/?tag=javamysqlanta-20) book, which contains 150 programming questions and solutions, good enough to do well on any programming job interviews e.g. Java, C++, Python or Ruby.

**Logic of Solution 1 - finding duplicates in O(n^2)**

In the first solution, we compare each element of the array to every other element. If it matches then its duplicate and if it doesn't then there are no duplicates. This is also known as **brute force algorithm** to find duplicate objects from Java array. The time complexity of this problem is O(n^2) or quadratic. When you give this solution to your interviewer, he will surely ask you to come up with O(n) time complexity algorithm, which we will see next.  
  
  
Here is the code to [find duplicate elements using brute force](http://javarevisited.blogspot.com/2015/06/3-ways-to-find-duplicate-elements-in-array-java.html) algorithm in Java:

public static Set<Integer> findDuplicates(int[] input) {

Set<Integer> duplicates = new HashSet<Integer>();

for (int i = 0; i < input.length; i++) {

for (int j = 1; j < input.length; j++) {

if (input[i] == input[j] && i != j) {

// duplicate element found

duplicates.add(input[i]);

break;

}

}

}

return duplicates;

}

Here instead of [printing the duplicate elements](http://javarevisited.blogspot.com/2012/02/how-to-check-or-detect-duplicate.html), we have stored them in a Set and returned from the method, but if Interviewer doesn't ask you to return duplicates then you can simply print them into console as I have done in next solution.

**Logic of Solution 2 - finding duplicates in O(n)**

Second solution demonstrate how you can use a suitable data structure to come up with better algorithm to solve the same problem. If you know, in Java, the Set interface doesn't allow duplicates and its based upon hash table data structure so insertion take O(1) time in average case. By using [HashSet](http://java67.blogspot.com/2015/09/how-to-sort-hashset-in-java-example.html" \t "_blank), a general purpose Set implementation, we can find duplicates in O(n) time. All you need to do is iterate over array using [advanced for loop](http://java67.blogspot.com/2012/08/how-to-traverse-iterate-or-loop-ArrayList-in-java-example-tutorial.html) and insert every element into HashSet. Since it allows only unique elements, add() method will fail and return false when you try to add duplicates. Bingo, you have find the duplicate element, just print them off to console, as shown in following program:

public static <T extends Comparable<T>> void getDuplicates(T[] array) {

Set<T> dupes = new HashSet<T>();

for (T i : array) {

if (!dupes.add(i)) {

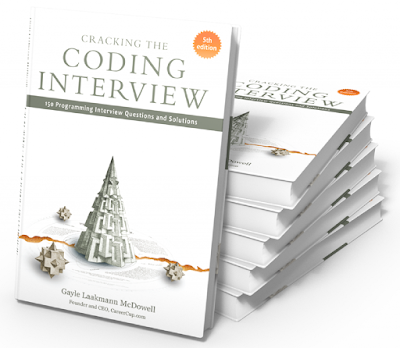
System.out.println("Duplicate element in array is : " + i);

}

}

}

This solution also demonstrate how you can [use Generics to write type-safe code](http://javarevisited.blogspot.com/2011/09/generics-java-example-tutorial.html) in Java. This method will work on any type of Java array e.g. Array with Integer, Array with String or any object which implements Comparable interface, but will not work with primitive array because they are not object in Java.

[](http://www.amazon.com/dp/098478280X/?tag=javamysqlanta-20)

**Java Program to find duplicate elements in Java using Generics**  
Here is the Java program to combine both solution, you can try running this solution on Eclipse IDE and see how it works. You can also write JUnit test to see our solution work in all cases especially corner cases like empty array, array with null etc.

import java.util.Arrays;

import java.util.HashSet;

import java.util.Set;

import static java.lang.System.\*;

/\*\*

\* Java Program to find duplicate elements in an array. In this program, you

\* will learn two solution to find duplicate elements in integer array e.g.

\* brute force, by using HashSet data structure.

\*

\* @author java67

\*/

public class DuplicatesFromArray{

public static void main(String args[]) {

int[] withDuplicates = { 1, 2, 3, 1, 2, 3, 4, 5, 3, 6 };

Set<Integer> duplicates = findDuplicates(withDuplicates);

out.println("input array is : " + Arrays.toString(withDuplicates));

out.println("Duplicate elements found in array are : " + duplicates);

// now calling our generic method to find duplicates

String[] myArray = { "ab", "cd", "ab", "de", "cd" };

out.println("input string array is : " + Arrays.toString(myArray));

getDuplicates(myArray);

}

/\*\*

\* Complexity of this solution is O(n^2)

\*

\* @param input

\* @return

\*/

public static Set<Integer> findDuplicates(int[] input) {

Set<Integer> duplicates = new HashSet<Integer>();

for (int i = 0; i < input.length; i++) {

for (int j = 1; j < input.length; j++) {

if (input[i] == input[j] && i != j) {

// duplicate element found

duplicates.add(input[i]);

break;

}

}

}

return duplicates;

}

/\*\*

\* Generic method to find duplicates in array. Complexity of this method is

\* O(n) because we are using HashSet data structure.

\*

\* @param array

\* @return

\*/

public static <T extends Comparable<T>> void getDuplicates(T[] array) {

Set<T> dupes = new HashSet<T>();

for (T i : array) {

if (!dupes.add(i)) {

System.out.println("Duplicate element in array is : " + i);

}

}

}

}

Output :

input array is : [1, 2, 3, 1, 2, 3, 4, 5, 3, 6]

Duplicate elements found in array are : [1, 2, 3]

input string array is : [ab, cd, ab, de, cd]

Duplicate element in array is : ab

Duplicate element in array is : cd

That's all about **how to find duplicate elements in an array**. You have now learned two ways to solve this problem in Java. First solution is the brute force algorithm which is demonstrate by finding duplicate element on integer array, but you can use the logic to find duplicate on any kind of array. Second solution uses HashSet data structure to reduce the time complexity from O(n^2) to O(n) and it also shows you can write generic methods to find duplicates on any object array.

Read more: <https://www.java67.com/2015/10/2-ways-to-find-duplicate-elements-in-java-array.html#ixzz6GexcKdyx>